UNIT 10 KOHONEN NETWORKS

Structure **Page No.**

10.1 INTRODUCTION

Typical neural network tries to simulate the biological human brain. Explaining how the human brain learns is quite difficult for a simple reason: nobody knows it exactly. However, it is known that human brain consists of a large number of neural cells that process information. Each cell works like a simple processor and only the massive interaction between all cells and their parallel processing makes the brain's abilities possible. If a certain amount of electrical stimulation is received by a neuron, it generates an output to all other connected neurons. Now if the incoming stimulation is too low, no output is generated by the neuron and the information's further transport will be blocked. On the other hand, if a certain amount of stimulation is received by a neuron, it generates an output to all other connected neurons and so information takes its way to its destination where some reaction will occur. It is supposed that during the learning process the connection structure among the neurons is changed, so that certain stimulations are only accepted by certain neurons. This means that the firm connections exist between the neural cells that once have learned a specific fact, enabling the fast recall of this information. Unlike the biological model, a neural net has an unchangeable structure. It is built of a specified number of neurons and a specified number of connections between them–*weights*–which have certain values.

The Kohonen network is named after the Finnish researcher, Teuvo Kohonen from University of Helsinki who pioneered the research and first presented in 1982. A Kohonen network is a two-layered network, much like the Perceptron. But the output layer for a two-neurone input layer can be represented as a two-dimensional grid, also known as the "competitive layer", which we shall discuss in Sec. 10.2.

To the supervised learning algorithms the desired outputs are provided for the comparison with the actual outputs in order to measure performance of the systems or the error caused by the system. The difference between actual output and target, an error value is also used for adjusting the weights between neurons. It is known that the *cortex* of the human brain is subdivided in different regions, each responsible for certain functions. To simulate this model of human brain, neural cells are geared to organize themselves in groups, according to incoming information. The Kohonen network is most famous neural network model designed in accordance to this type of *unsupervised learning*. Here the desired output is not provided during the modeling phase, therefore Kohonen Networks function based on the principles of *competitive learning* or *self-organization*. We shall discuss Kohonen networks in Secs. 10.3, 10.4 and 10.5 and self organizing feature map in Sec.10.6.

Objectives

After reading this unit will be able to:

- clarify about different types Artificial Neural Networks;
- differentiate Kohonen networks from other feed forward back propagation neural networks;
- define and explain the fundamentals, the types, the basic algorithm of Kohonen networks;
- explain the terms unsupervised learning, competitive learning, SOM, etc;
- apply the Kohonen Networks, SOM for the kinds of learning problems they were designed.

10.2 COMPETITIVE LEARNING

Competitive learning: *When one student achieves the goal all other students fail to reach that goal.*

A basic **competitive learning** network has one layer of input neurons and one layer of output neurons. An input pattern x is a sample point in the n-dimensional real or binary vector space. Binary-valued (1 or 0) *local representations* are more often used for the output nodes. That is, there are as many output neurons as the number of classes and each output node represents a pattern category.

A competitive learning network comprises the feed forward excitatory network(s) and the lateral inhibitory network(s). The feed forward network usually implements an excitatory **Hebbian learning rule**. It consists of when an input cell persistently participates in firing an output cell, the input cell's influence firing that output cell is increased. The lateral competitive network is inhibitory in nature. The network serves the important role of selecting the winner, often via a competitive learning process, highlighting the **"winner-take-all"** schema. In a winner-take-all circuit, the output unit receiving the largest input is assigned a full value (e.g. 1), whereas all other units are suppressed to a 0 value. The winner-take-all circuit is usually implemented by a (digital or analog) MAXNET network. Another example of a lateral network is Kohonen's self-organizing feature map. By allowing the output nodes to interact via the lateral network, the neural model can be trained to preserve certain topological ordering.

Using no supervision from any teacher, unsupervised networks adapt the weights and verify the results only on the input patterns. One popular scheme for such adaptation is the competitive learning rule, which allows the units to compete for the exclusive right to respond to a particular input pattern. It can be viewd as a sophisticated clustering technique, whose objective is to divide a set of input patterns into a number of clusters such that the patterns of the same cluster exhibit a certain degree of similarity. The training rules are often the Hebbian rule for the feed forward network and the winner-take-all (WTA) rule for the lateral network.

Goals of Competitive Learning

1. **Error Minimization** – Minimizing the quantization (distortion) errors. A typical application where error minimization is important is *vector quantization*. In vector quantization data is transmitted over limited bandwidth communication channels by transmitting for each data vector only the *index* of the nearest reference vector. The set of reference vectors (which is called *codebook* in this context) is assumed to be known both to sender and receiver. Therefore, the receiver can use the transmitted indexes to retrieve the

corresponding reference vector. There is an information loss in this case which is equal to the distance of current data vector and nearest reference vector. The expectation value of this error is described by equations and in particular if the data distribution is clustered (contains sub regions of high probability density), dramatic compression rates can be achieved with vector quantization with relatively little distortion.

2. **Entropy maximization** – Sometimes the reference vectors should be distributed such that each reference vector has the same chance to be winner for a randomly generated input signal ξ:

$$P(s(\xi) = c) = \frac{1}{|A|} \quad (\forall c \in A)$$

An advantage of choosing reference vectors such as to maximize entropy is the inherent robustness of the resulting system. The removal (or ``failure'') of any reference vector affects only a limited fraction of the data.

3. **Feature Mapping** – With some network architectures it is possible to map high-dimensional input signals onto a lower-dimensional structure in such a way, that some similarity relations present in the original data are still present after the mapping. This is denoted by *feature mapping* and can be useful for data visualization. A prerequisite for this is that the network used has a fixed dimensionality. This is the case, e.g., for the *self-organizing feature map* and the other methods discussed in section of this report. A related question is, how *topology-preserving* is the mapping from the input data space onto the discrete network structure, i.e., how well are similarities preserved? Several quantitative measures have been proposed to evaluate this like the topographic product or the topographic function.

4. **Other** – Clustering, density estimation, combining with supervised learning (RBF) obtain better results.

Now, try the following exercise.

E1) Write any two goals of competitive learning.

Now, let us begin the following section by defining the Kohonen networks.

10.3 KOHONEN NETWORKS

The objective of a Kohonen network is to map input vectors (patterns) of arbitrary dimension N onto a discrete map with 1 or 2 dimensions. Patterns close to one another in the input space should be close to one another in the map: they should be topologically ordered. A Kohonen network is composed of a grid of output units and N input units. The input pattern is fed to each output unit. The input lines to each output unit are weighted. These weights are initialized to small random numbers.

Distinctiveness of Kohonen Networks

The Kohonen neural network differs from the usual feed forward back propagation neural network not in architecture but in its functionality. The Kohonen neural network is different both in how it is trained and how it recalls a pattern. The Kohonen neural network does not use any activation function and also a bias weight. The output from the Kohonen neural network is not a composition of the output of several neurons. On input of a pattern to a Kohonen network one of the output neurons is selected as a "winner". This "winning" neuron is the output from the

Kohonen network. Often these "winning" neurons represent groups in the data that is presented to the Kohonen network.

The most significant difference between the Kohonen neural network and the feed forward back propagation neural network is that the Kohonen network trained in an unsupervised mode. This means that the Kohonen network is presented with data, but the correct output that corresponds to that data is not specified. Using the Kohonen network this data can be classified into groups. We will begin our review of the Kohonen network by examining the training process.

It is also important to understand the limitations of the Kohonen neural network. You will recall from the previous unit that neural networks with only two layers can only be applied to linearly separable problems. This is the case with the Kohonen neural network. Kohonen neural networks are used because they are a relatively simple network to construct that can be trained very rapidly.

Basic Algorithm

The algorithm for training a Kohonen network can be summarized in the following steps:

Step 1: Apply an input vector X to the network.

Step 2: Calculate the distance D_j (in n-dimensional space) between X and the weight vectors W_j of each neuron.

Step 3: The neuron that has the weight vector closest to X is declared the winner. Use this weight vector W_c as the centre of a group of weight vectors that lie within a distance of d from W_c.

Step 4: Train this group of vectors according to for all weight vectors within a distance d of W_c.

Step 5: Perform steps 1 through 4 for each input vector.

The n connection weights into a neuron are treated as a vector in n-dimensional space. Before training, the vector is initialized with random values, and then the values are normalized to make the vector of unit length in weight space. The input vectors in the training set are likewise normalized. As training proceeds, the values of d are gradually reduced. It is recommended by Kohonen that start near 1 and reduce to 0.1, whereas d can start as large as the greatest distance between neurons and reduce to a single neuron.

Now try the following exercise.

E2) Differentiate Kohonen networks for the feed forward and back propagation neural network.

Now, in the following section, we shall discuss the structure of Kohonen networks in detail.

10.4 STRUCTURE OF KOHONEN NETWORKS

The Kohonen neural network works differently than the feed forward neural network. The Kohonen neural network contains only an input and output layer of neurons.

There is no hidden layer in a Kohonen neural network. First we will examine the input and output to a Kohonen neural network.

The input to a Kohonen neural network is given to the neural network using the input neurons. These input neurons are each given the floating point numbers that make up the input pattern to the network. A Kohonen neural network requires that these inputs be normalized to the range between -1 and 1. Presenting an input pattern to the network will cause a reaction from the output neurons.

The output of a Kohonen neural network is very different from the output of a feed forward neural network. In a typical feed-forward network with five output neurons the number of output values is consisted to be five. However, in the case of a Kohonen neural network only one of the five output neurons actually produces a value. Additionally, this single value is either true or false. When the pattern is presented to the Kohonen neural network, one single output neuron is chosen as the output neuron. Therefore, the output from the Kohonen neural network is usually the index of the neuron (i.e. Neuron $\#5$) that fired. The structure of a typical Kohonen neural network is shown in Fig. 1.

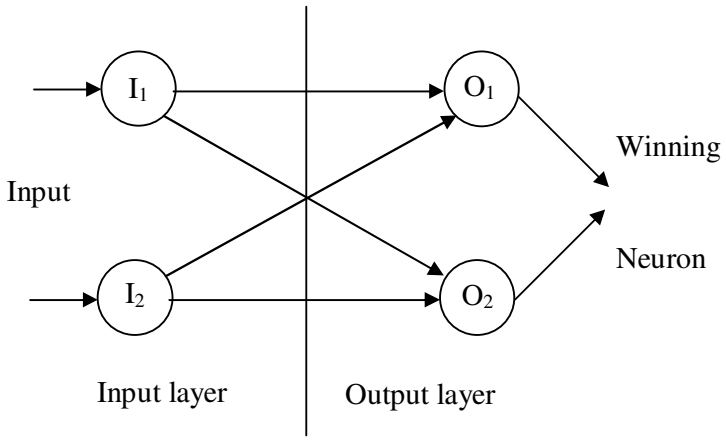


Fig. 1: A Kohonen Neural Network

Example 1: Consider a very simple Kohonen neural network. This network includes only two input and two output neurons. The input given to the two input neurons is shown in Table 10.1.

Table 10.1: Inputs to a Kohonen Neural Network

Input Neuron 1 (I_1)	0.5
Input Neuron 2 (I_2)	0.75

Consider the connection weights between the neurons as given in Table 10.2.

Table 10.2: Connection weights in the sample Kohonen neural network

$I_1 -> O_1$	0.1
$I_2 -> O_1$	0.2
$I_1 -> O_2$	0.3
$I_2 -> O_2$	0.4

Using these values we examine which neuron would win and produce output. This can be achieved by normalizing the input.

Normalization of Inputs

The Kohonen neural network also requires that its input be normalized. Kohonen neural network is considered to have a two layer network because there are only two actual neuron layers at work in the Kohonen neural network.

The requirements that the Kohonen neural network places on its input data are one of the most sever limitations of the Kohonen neural network. Input to the Kohonen neural network should be between the values -1 and 1. In addition, each of the inputs should fully use the range. If one, or more, of the input neurons were to use only the numbers between 0 and 1, the performance of the neural network would suffer.
To normalize the input we must first calculate the "vector length" of the input data, or vector. This is done by summing the squares of the input vector.

Example 2: Normalize the input given in Example 1.

Solution: Normalization of input gives us $= (0.5*0.5)+(0.75*0.75)$
This would result in a "vector length" of 0.8125. If the length becomes too small, say less than the length is set to that same arbitrarily small value. In this case the "vector length" is a sufficiently large number. Using this length we can now determine the normalization factor. The normalization factor is the reciprocal of the square root of the length. For our value the normalization factor is calculated as follows.

$$\frac{1}{\sqrt{0.8125}}.$$

This results in a normalization factor of 1.1094. This normalization process will be used in the next step where the output layer is calculated.

Calculating Each Neuron's Output

To calculate the output the input vector and neuron connection weights must both be considered. First the "dot product" of the input neurons and their connection weights must be calculated. To calculate the dot product between two vectors you must multiply each of the elements in the two vectors. We will now examine how this is done.

The Kohonen algorithm specifies that we must take the dot product of the input vector and the weights between the input neurons and the output neurons.

Example 3: Calculate the output of each of the neuron of Example 1.

Solution: The output from the neuron 1 is

$$|0.5\ \ 0.75|\bullet|0.1\ \ 0.2| = (0.5*0.75)+(0.1*0.2)\ = 0.395$$

The output from the neuron 2 is $|0.5\ \ 0.75|\bullet|0.3\ \ 0.4| = 0.495$

In the following section, we shall discuss the methods for modifying the weights.

10.5 METHODS FOR WEIGHT UPDATES

There are several methods for updating a neuron's weight vector in response to input pattern. The original algorithm proposed by Kohonen is to add a fraction of the input

vector to the weight vector, renormalizing the sum vector after that. This pushes the weight vector in the direction of the data vector "bit by bit". If x is the input vector presented to the network, and w^t presents the weight vector of the winning neuron at time t, then the updated weight vector w^{t+1} can be calculated as,

$$w^{t+1} = w^t = \alpha x / | w^t + \alpha x |$$

Denominator in this equation calculates the length (Euclidean norm) of the vector, and the constant α is called the *learning rate*. It is always much less than 1, and is usually decreased as training progresses.

Another, subtractive method relies on the fact that a weight vector can be pushed toward input data vector by subtracting them (thus finding the error difference), and adding the fraction of this difference to the weight vector.

$$\varepsilon = x - w^t$$

$$w^{t+1} = w^t + \alpha\varepsilon$$

There are many fine details that should be studied in order to successfully use this type of neural network in your projects. These include *conscience* mechanism, *inhibition* algorithms, etc. Below are some excellent sites that you can use to gain more knowledge on this subject.

Error Calculation

Unlike the case of supervised learning where the anticipated output of the neural network before training the network is available for comparing it with the actual output. Therefore the error is computed as the difference between the anticipated output of the neural network and the actual output of the neural network. However, in case of unsupervised training there is no anticipated output. Therefore the calculated error is not the true error, or at least not an error according to the normal understanding.

The purpose of the Kohonen neural network is to classify the input into several sets. The error for the Kohonen neural network, therefore, must be able to measure how well the network is classifying these items. Two methods for determining the error in are examined in this section. There is no official way to calculate the error for a Kohonen neural network. The error is just a percent number that gives an idea of how well the Kohonen network is classifying the input into the output groups. The error itself is not used to modify the weights, as was done in the back propagation algorithm.

Now, let us discuss the self-organizing feature map in the following section.

10.6 SELF-ORGANIZING FEATURE MAP

Kohonen networks are based on the concept of competitive learning. However, based on their application for various purposes Kohonen networks are further categorized in two types: Self-organizing Feature Maps and Vector Quantization. Here we elaborate upon the most popularly used Kohonen model i.e. Self-Organizing Maps (SOM).

Self-organizing system is a special class of ANN based on competitive learning. The input layer contains n neurons to input the pattern while in the output layer the neurons of the network compete among themselves to be activated or fired with the result that only one output neuron is on at any one time. The output neuron that wins the competition is called a winner-takes-all or simply the winning neuron. One way of inducing a winner-take-all competition is to use the lateral inhibitory connections i.e. negative feed-back paths between them.

In a self-organizing map, the neurons are placed at the nodes of a lattice that is whether one- or two-dimensional. The neurons become tuned to various input patterns or the classes of the input patterns in the course of a competitive learning process. The location of the winning neuron becomes ordered with respect to each other in such a way that a meaningful coordinate system for different input features is created over the lattice. A self-organizing map is therefore characterized by the formation of a topographic map of the input patterns in which the coordinates (spatial locations) of the neurons in the lattice are indicative of the statistical features contained in the input patterns. The self-organization maps are inherently non-linear. This class of neural network model is motivated by the features of the human brain. Human brain is organized in many places in such a way that different sensory inputs are represented by topologically ordered computational maps. For example, sensory inputs such as visual, acoustic, tactile, etc. are mapped onto different areas of the cerebral cortex in a topologically ordered manner. A computational map is defined by an array of neurons representing slightly differently tuned processors which operate on the sensory information bearing signals in parallel. Thus, the neurons are expected to transform input signals into a place coded probability distribution that represents the computed values of parameters by sites of relative activity within the map.

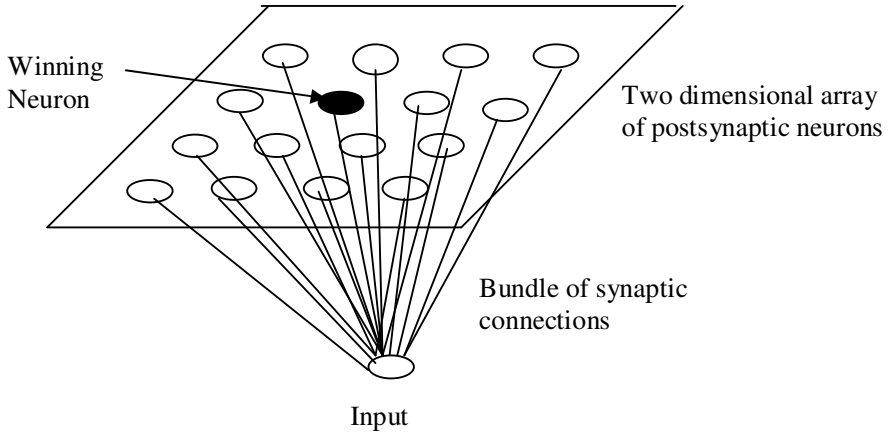


Fig. 2: Basic model of SOM

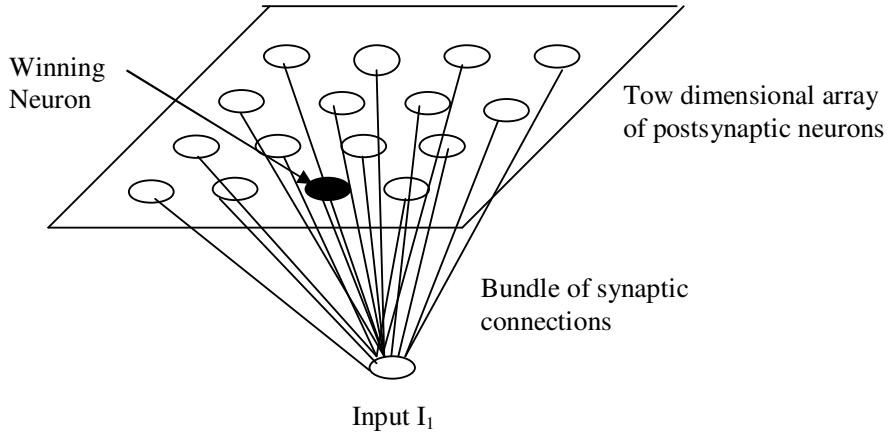


Fig. 3: SOM for an input pattern I_1

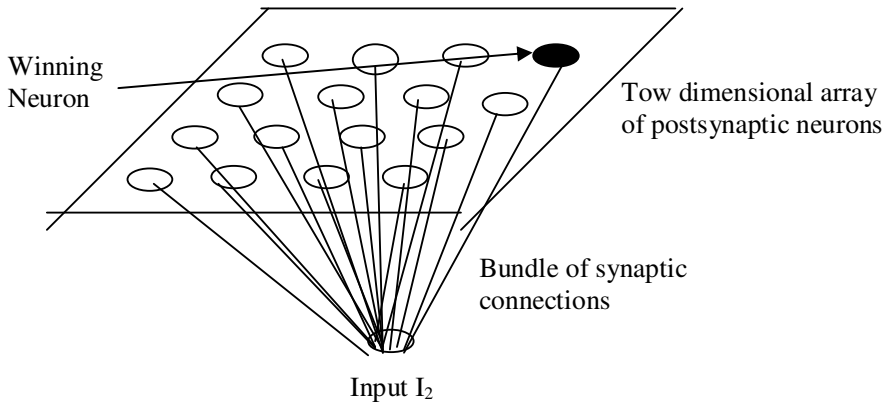


Fig. 4: SOM for an input pattern I_2

Fig. 2 presents the basic models for feature mapping as proposed by Kohonen while Fig. 3 and Fig. 4 exhibit the winning neuron to be different in case the input patterns I_1 and I_2 are not similar. The use of computational maps offers the following properties:

1. At each stage of representation, each incoming piece of information is kept in its proper context,
2. Neurons dealing with closely related pieces of information are close together so that they can interact via short synaptic connections.

From the design of self-organizing maps using computational maps the following principle of topographic map formulation was proposed by Kohonen:

The spatial location of an output neuron in a topographic map corresponds to a particular domain or feature of data drawn from the input space.

The principle goal of the SOM is to transfer an incoming signal pattern to arbitrary dimension into a one- or two-dimensional discrete map and to do this transformation adaptively in a topologically ordered fashion. The above model of SOM captures the essential features of computational maps in the brain and yet remains computationally tractable. This model also belongs to the class of vector-coding algorithms. It provides a topological mapping that optionally places a fixed number of vectors (code words) into a higher dimensional input space and thereby facilitates data compression.

The algorithm for the training of the self-organizing maps has two fundamental steps. The first step is for the initializing the synaptic weight of the network and the second step is for the training. Initialization is done by assigning small values picked from a random number generator. For doing this no prior order is imposed on the feature map. After initialization, in the training step the following three processes are executed:

1. Competition,
2. Cooperation, and
3. Synaptic Adaptation

Competition: For each input pattern, the neuron in the network compute their respective values of a discriminate function. This discriminate function provides the basis for competition among the neurons. The neuron with the largest value of discriminate function is declared winner of the competition.

Cooperation: The winning neuron determines the spatial location of a topological neighbourhood of excited neurons. Thus, it provides the basis for cooperation among the neighbouring neurons.

Synaptic Adaptation: This process enables the excited neuron to increase its individual value of the discriminate function in relation to the input pattern through suitable adjustments applied to their synaptic weights. The adjustments made are influenced by the response of the winning neuron to a subsequent application of a similar input pattern in an enhanced manner.

Example 4 (A Graphic Example):

Here we illustrate the self-organisation of a Kohonen net graphically. Consider an input space with two components only. Let the lattice include 6 nodes on a rectangular grid where the winner neuron will be excited as output.

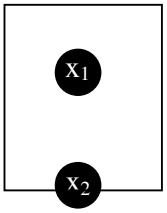


Fig. 5: Input neurons

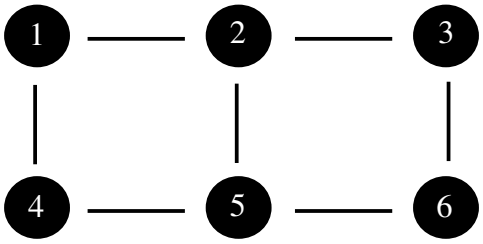


Fig. 6: Kohonen net of 6 neurons net on grid (output lattice)

Since the input space having only two components each of the neurons on the lattice will be initialized with randomly assigned/selected two weights. Thus another representation of the Kohonen net (lattice) may be in the *weight space* as presented in the Fig. 7. The lines denote the connection between the adjacent neurons in the initial output lattice as in Fig. 6.

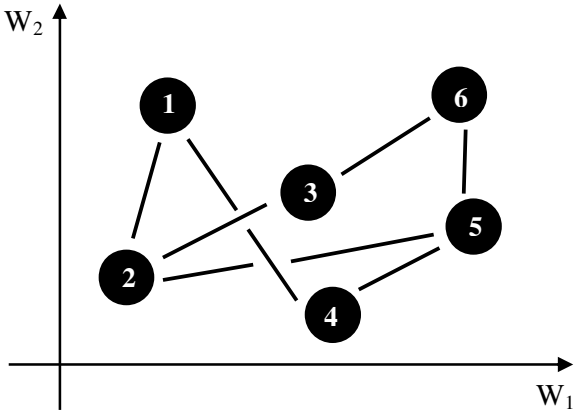


Fig. 7: Output lattice of Fig. 6 in weight space

Input Patterns: Suppose now that there are 6 input patterns (vectors). The representation of the patterns A, B, C, D, E and F corresponding the two components x_1 and x_2 is as shown below.

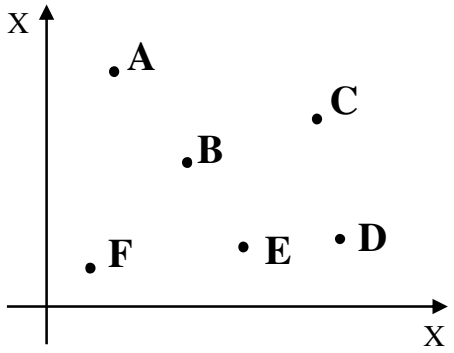


Fig. 8: Input patterns in two component space

In a well trained (ordered) net that has developed a topographic map the diagram in weight space should have the same topology as that in physical space and therefore should reflect the properties of the training set.

Trained weight space

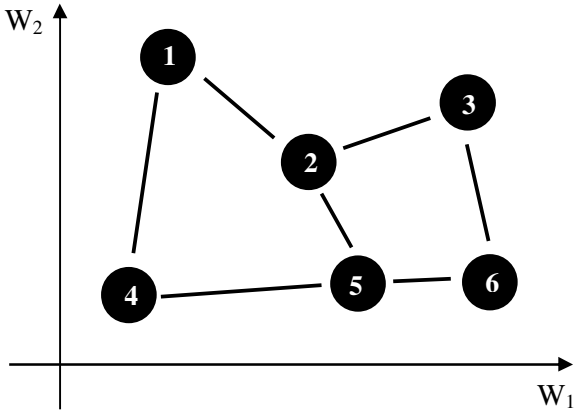
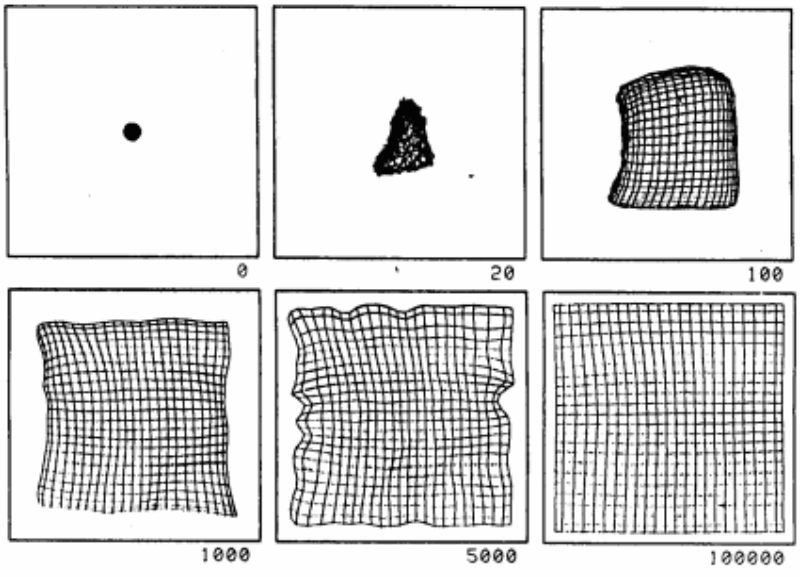


Fig. 9: Neurons trained based on the 6 input patterns A, B, C, D, E and F

Example 5: Self-Organizing Feature Maps



The case of 2-component vectors which are drawn randomly from the unit square and in which the initial weights are close to the centre of the unit square, is dealt with by Kohonen as in the above picture.

The weight diagram starts as a 'crumpled ball' at the centre and expands like fishing net being unraveled. Kohonen deals with several other similar examples.

When the input space is more than 2-dimensional, the higher dimensions have to get 'squashed' onto the grid. This will be done in such a way as to preserve the most important variations in the input data. Thus, it is often the case that the underlying dimensionality of the input space is smaller than the number of input patterns.

Example 6: If the input vector are $I_1 = [-1\ 0]^T, I_2 = [0\ 1]^T$ and $I_3 = [\sqrt{2}\ \ 1/\sqrt{2}]^T$ and initial values of three weight vectors are $[0\ -1]^T, [-2/\sqrt{5}\ \ 1/\sqrt{5}]^T$, $[-1/\sqrt{5}\ \ 2/\sqrt{5}]^T$, calculate the resulting weight found after training the competitive layer with Kohonen's rule and a learning rate α of 0.5 on the input-series in order I_1, I_2 and I_3.

Solution: First combining weight-vectors into a weight matrix

$$W = \begin{bmatrix} 0 & -1 \\ -2/\sqrt{5} & -1/\sqrt{5} \\ -1/\sqrt{5} & 2/\sqrt{5} \end{bmatrix}$$

Now presenting first-input vector I_1:

$$a = f(W.I_1) = f\left(\begin{bmatrix} 0 & -1 \\ -2/\sqrt{5} & 1/\sqrt{5} \\ 1/\sqrt{5} & 2/\sqrt{5} \end{bmatrix}\begin{Bmatrix} -1 \\ 0 \end{Bmatrix}\right) = f\left(\begin{bmatrix} 0 \\ 0.894 \\ 0.447 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

(As the function f ( ) is competitive function.)
The second neuron responded, since $W^{(2)}$ is closest to I_1, so update $W^{(2)}$ with Kohonen's rule:

$$\begin{aligned} W_{new}^{(2)} &= W_{old}^{(2)} + \alpha\left(I_1 - W_{old}^{(2)}\right) \\ &= \begin{bmatrix} -2/\sqrt{5} \\ 1/\sqrt{5} \end{bmatrix} + 0.5\left(\begin{bmatrix} -1 \\ 0 \end{bmatrix} - \begin{bmatrix} -2/\sqrt{5} \\ 1/\sqrt{5} \end{bmatrix}\right) \\ &= \begin{bmatrix} -0.947 \\ 0.224 \end{bmatrix} \end{aligned}$$

Now present second input I_2:

$$a = f\left(W.I_2\right) = f\left(\begin{bmatrix} 0 & -1 \\ -0.947 & 0.224 \\ -1/\sqrt{5} & 2/\sqrt{5} \end{bmatrix}\begin{bmatrix} -1 \\ 0 \end{bmatrix}\right)$$

$$= f\left(\begin{bmatrix} -1 \\ 0.224 \\ 0.894 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Here third neuron won, so its weights move closer to I_2 :

$$W_{new}^{(3)} = W_{old}^{(3)} + \alpha\left(I_1 - W_{old}^{(3)}\right)$$

$$= \begin{bmatrix} -2/\sqrt{5} \\ 1/\sqrt{5} \end{bmatrix} + 0.5\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} -1/\sqrt{5} \\ 2 \end{bmatrix}\right) = \begin{bmatrix} -0.224 \\ 0.947 \end{bmatrix}$$

Now present I_3 :

$$a = f\left(W.I_3\right) = f\left(\begin{bmatrix} 0 & -1 \\ -0.947 & 0.224 \\ -0.224 & 0.947 \end{bmatrix}\begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}\right)$$

$$= f\left(\begin{bmatrix} -0.707 \\ -0.512 \\ 0.512 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Third neuron won again:

$$W_{new}^{(3)} = W_{old}^{(3)} + \alpha\left(I_3 - W_{old}^{(3)}\right)$$

$$= \begin{bmatrix} 0.224 \\ 0.947 \end{bmatrix} + 0.5\left(\begin{bmatrix} -1/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix} - \begin{bmatrix} -0.224 \\ 0.947 \end{bmatrix}\right)$$

$$= \begin{bmatrix} -0.2417 \\ 0.8272 \end{bmatrix}$$

The final structure is now $W = \begin{bmatrix} 0 & -1 \\ -0.947 & 0.224 \\ 0.2417 & 0.8272 \end{bmatrix}$

Now try the following exercises.

E3) Train a competitive network using the following input pattern:
$I_1 = \begin{bmatrix} 1 & -1 \end{bmatrix}^T, I_2 = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$ and $I_3 = \begin{bmatrix} -1 & 1 \end{bmatrix}^T$. Use the Kohonen learning law with $\alpha = 0.5$ and train for one pass through the input patterns (Present each input once in order given.) Assume an initial weight matrix as $\begin{bmatrix} \sqrt{2} & 0 \\ 0 & \sqrt{2} \end{bmatrix}$.

Applications of Self-organizing Maps

1. The SOM is useful for vector quantization, clustering, feature extraction, and data visualization.
2. The SOM is well known for its ability to perform clustering while preserving topology.
3. The SOM was originally intended to approximate input signals or their PDFs, by quantified codebook vector that are localized in the input space to minimize a quantization error functional.
4. The SOM is able to divide the input space into region with common nearest reference vectors.
5. The SOM is related to adaptive C-means, but performs a topological feature map that is more complex than just cluster analysis.
6. The SOM is especially powerful for the visualization of high-dimensional data.
7. The SOM can be used to decompose complex information-processing systems into a set of simple subsystems.

Now, let us summarize the unit.

10.7 SUMMARY

In this unit, we have covered the following points.

1. The concepts and principles of competitive learning, which is the basis of Kohonen model of artificial neural networks were discussed. A list of the goals of the competitive learning gives a broader understanding of error minimization, entropy maximization, features maps, clustering, density estimation, etc.
2. The major part of this unit presents various aspects of Kohonen Network, namely the structure, normalization of the input, weight initialisation and weight updates and finally the error calculation.
3. Some attention is also devoted to Self-Organization feature Maps (SOM) in order to give you idea of the processes of Competition, Cooperation, and Synaptic Adaptation, which operate in SOM. Finally one graphical example of Kohonen network map is presented for a clearer understanding of the this class of network.

10.8 SOLUTIONS/ANSWERS

E1) Any two goals described in Sec. 10.2 may be given.

E2) Kohenen network is trained in unsupervised mode and is presented with data but the correct output that corresponds to that data is not specified. This differentiates the kohonen network from the feed forward backward propagation neural networks.

10.9 PRACTICAL ASSIGNMENT

Session 9

Write a program in ‘C’ language to find the updated/ modified weights for Kohonen Networks. Also test your program on the input patterns given in Example 6.

REFERENCE

1. Neural Networks in a Soft Computing Framework, K.-L.DU, M.N.S. Swamy.
2. Soft Computing, D.K. Pratihar.
3. Neuro-Fuzzy and Soft Computing, J.S.R. Jang, C.-T. Sun, E. Mizutani.
4. Neural Network, M. Ananda Rao, J. Srinvas.
5. Neural Network! A Classroom Approach, Satish, Tata McGraw-Hill.
6. http://highered.mcgraw_hill.com/site/0070482926.
7. Discovering knowledge in data – an introduction in data mining.
8. www.rgu.ac.ux/files/chapter7-hopfield.pdf
9. www.comp.leeds.ac.yx/az23/reading/Hopfield.pdf.